


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☐ The ACM Digital Library ☒ The Guide

THE GUIDE TO COMPUTING LITERATURE


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

 Terms used **expression temporary**

Found 8 of 869,947

Sort results by

☒ [Save results to a Binder](#)
[Try an Advanced Search](#)

Display results

☒ [Search Tips](#)

 Try this search in [The Digital Library](#)
☐ Open results in a new window

Results 1 - 8 of 8

 Relevance scale ☐ ☐ ☐ ☐ ☐

1 [Partial redundancy elimination in SSA form](#)



Robert Kennedy, Sun Chan, Shin-Ming Liu, Raymond Lo, Peng Tu, Fred Chow

 May 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 3

 Full text available: [pdf\(704.71 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citings](#), [index terms](#)

The SSAPRE algorithm for performing partial redundancy elimination based entirely on SSA form is presented. The algorithm is formulated based on a new conceptual framework, the factored redundancy graph, for analyzing redundancy, and represents the first sparse approach to the classical problem and on methods for its solution. With the algorithm description, theorems and their proofs are given showing that the algorithm produces the best possible code by the criteria of computational optim ...

Keywords: code motion, common subexpressions, data flow analysis, partial redundancy, static single assignment form

2 [Reflection in an object-oriented concurrent language](#)



Takuo Watanabe, Akinori Yonezawa

 January 1988 **ACM SIGPLAN Notices, Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 23 Issue 11

 Full text available: [pdf\(1.19 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citings](#), [index terms](#)

Our work is along the line of the work of B. Smith and P. Maes. We first discuss our notion of reflection in object-oriented concurrent computation and then present a reflective object-oriented concurrent language ABCL/R. We give several illustrative examples of reflective programming such as (1) dynamic concurrent acquisition of "methods" from other objects, (2) monitoring the behavior of concurrently running objects, and (3) augmentation of th ...

3 [Introduction to the Special Issue on Adaptive Workflow Systems](#)



Mark Klein, Chrysanthos Dellarocas, Abraham Bernstein

 August 2000 **Computer Supported Cooperative Work**, Volume 9 Issue 3-4

 Full text available: [Publisher Site](#) Additional Information: [full citation](#), [citings](#)

4 [Affix grammar driven code generation](#)



Mahadevan Ganapathi, Charles N. Fischer

 October 1985 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 7 Issue 4



[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

Search: ☐ The ACM Digital Library ☒ The Guide

finalization block exception

SEARCH

THE GUIDE TO COMPUTING LITERATURE



[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used finalization block exception

Found 26,313 of 869,947

Sort results
by .

relevance

 Save results to a Binder

Try an Advanced Search

Try this search in The Digital Library

Display results

expanded form



Search Tips

☐ Open results in a new window

Results 1 - 20 of 200

Result page: **1** [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)


Relevance scale ☐ ☒ ☐ ☐ ☐

Best 200 shown

1 The finalization operation for abstract types

Richard L. Schwartz, P. M. Melliar-Smith

March 1981 **Proceedings of the 5th international conference on Software engineering**

Full text available:  pdf(769.32 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

In this paper we argue the importance of a finalization capability in a programming language abstract type facility. Finalization, the dual of initialization, is crucial for applications involving the allocation of, and access to, abstract resources. A semantic model for finalization is given, defining both statically and dynamically allocated abstract objects, in the presence of exception handling. For illustration, we incorporate finalization in an abstract data type facility designed as ...

2 A modular verifiable exception handling mechanism

Shaula Yemini, Daniel M. Berry

April 1985 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,

Volume 7 Issue 2

Full text available: pdf(2.38 MB)

This paper presents a new model for exception handling, called the replacement model. The replacement model, in contrast to other exception-handling proposals, supports all the handler responses of resumption, termination, retry, and exception propagation, within both statements and expressions, in a modular, simple, and uniform fashion. The model can be embedded in any expression-oriented language and can also be adapted to languages which are not expression oriented with almost all the ab ...

3 Part II: Articles: Implementing transactions using Ada exceptions: which features are missing?

M. Patiño-Martínez, R. Jiménez-Peris, S. Arévalo

September 2001 **ACM SIGAda Ada Letters**, Volume XXI Issue 3

Full text available: pdf(924.53 KB) Additional Information: [full citation](#), [abstract](#), [references](#)

Transactional Drago programming language is an *Ada* extension that provides transaction processing capabilities. Exceptions have been integrated with transactions in *Transactional Drago*; exceptions are used to notify transaction aborts and any unhandled exception aborts a transaction. Transactions can be multithreaded in *Transactional Drago*, and therefore, concurrent exceptions can be raised. In that case a single exception must be chosen to notify the transaction abort ...


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☐ The ACM Digital Library ☒ The Guide

THE GUIDE TO COMPUTING LITERATURE

[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

 Terms used hir exception

 Found **31,284** of **869,947**

Sort results by


[Save results to a Binder](#)
[Try an Advanced Search](#)
[Try this search in The Digital Library](#)

Display results


[Search Tips](#)
☐ Open results in a new window

Results 1 - 20 of 200

 Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

 Relevance scale ☐ ☐ ☐ ☐ ☐
1 [The Jalapeño dynamic optimizing compiler for Java](#)

Michael G. Burke, Jong-Deok Choi, Stephen Fink, David Grove, Michael Hind, Vivek Sarkar, Mauricio J. Serrano, V. C. Sreedhar, Harini Srinivasan, John Whaley

 June 1999 **Proceedings of the ACM 1999 conference on Java Grande**

 Full text available: [pdf\(1.34 MB\)](#)

 Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)
2 [Strength reduction for loop-invariant types](#)

Phung Hua Nguyen, Jingling Xue

 January 2004 **Proceedings of the 27th conference on Australasian computer science - Volume 26**

 Full text available: [pdf\(147.42 KB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#)

Types are fundamental for enforcing levels of abstraction in modern high-level programming languages and their lower-level representations. However, some type-related features such as dynamic method calls and dynamic type casts can contribute substantially to the performance of a program. Loop-invariant type is a concept relating to an object whose dynamic type never changes inside a loop. In this case, operations on the type of the object may be redundant in the loop. As these operations often ...

Keywords: PRE, inlining, loop-invariant type, strength reduction, type checking

3 [Efficient and precise modeling of exceptions for the analysis of Java programs](#)

Jong-Deok Choi, David Grove, Michael Hind, Vivek Sarkar

 September 1999 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 1999 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**, Volume 24 Issue 5

 Full text available: [pdf\(1.16 MB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

The Factored Control Flow Graph, *FCFG*, is a novel representation of a program's intraprocedural control flow, which is designed to efficiently support the analysis of programs written in languages, such as Java, that have frequently occurring operations whose execution may result in exceptional control flow. The *FCFG* is more compact than traditional CFG representations for exceptional control flow, yet there is no loss of precision in using the *FCFG*. In this paper, we introduce the *FCFG* r ...

4 [Data entry curricula guidelines: a working paper of the Community and Junior College](#)



**Software Patent Institute Database of Software
Technologies**

Search Results

Not signed on

Record Counts: EXCEPTION:13722 INTERMEDIATE:9198 REPRESENTATION:28316

New Search

Modify Search

Records 1 - 3 of 3

1. A. Description Facility [From A Machine Description Facility for Compiler Testing; V. MACHINE DESCRIPTION]

Publication Date: September 1, 1977

2. Affix Grammar Driven Code Generation

Publication Date: October, 1985

3. The Priority-Based Coloring Approach to Register Allocation

Publication Date: October, 1990

1

© Software Patent Institute, 2005


[Web](#) [Images](#) [Groups](#) [News](#) [Froogle](#) [Local](#) [more »](#)

[Advanced Search](#)
[Preferences](#)

Web

 Results 1 - 10 of about 684,000 for intermediate representation exception. (0.07 seconds)

Notes on IR

The original **intermediate representation** used by ARMphetamine ("Phetacode"), while just about ... UNDEF, Trigger undefined instruction **exception**, (none?) ...
 armphetamine.sourceforge.net/pheta2.html - 12k - [Cached](#) - [Similar pages](#)

[PDF] Harpoon Project Compiler **Intermediate Representation**

File Format: PDF/Adobe Acrobat - [View as HTML](#)
 ciliate analysis, **exception** handling and its associ- ... Figure 3: Classes comprising the QuadSSA **intermediate representation**. Only the constructors are ...
 www.flex-compiler.lcs.mit.edu/Harpoon/quads/quads.pdf - [Similar pages](#)

Intermediate Representation

Intermediate Representation. Cetus' IR contrasts with the Polaris Fortran ... whenever an **exception** is caught or the compiler terminates abnormally. ...
 paramount.www.ecn.purdue.edu/ParaMount/Cetus/manual/ch07s03.html - 13k - [Cached](#) - [Similar pages](#)

Java bytecode:

Bytecode is the **intermediate representation** of Java programs just as ... If an **exception** is generated while executing inside of a synchronized block (a ...
 www-106.ibm.com/developerworks/ibm/library/it-haggar_bytecode/ - 44k - [Cached](#) - [Similar pages](#)

[PPT] IPv6

File Format: Microsoft Powerpoint 97 - [View as HTML](#)
 bytecode is an **intermediate representation** of the program (class). ... throw an **exception**: signal that some condition (possibly an error) has occurred. ...
 www.cs.rpi.edu/~hollingd/netprog/notes/javaintro/javaintro.ppt - [Similar pages](#)

Comp.compilers: Re: (C as) **Intermediate Representation**

From comp.compilers newsgroup: Re: (C as) **Intermediate Representation**. ... **Exception**-handling can be implemented easily enough through use of ...
 compilers.iecc.com/comparch/article/90-08-046 - 11k - [Cached](#) - [Similar pages](#)

Trapp, Martin; Lindenmaier, Goetz; Boesler, Boris: Documentation ...

Documentation of the **Intermediate Representation** Firm Tech Report Nr. 1999-44 ... Firm does not express the ordering constraints imposed by the **exception** ...
 www.ubka.uni-karlsruhe.de/indexer-vvv/ira/1999/14 - 53k - [Cached](#) - [Similar pages](#)

[PPT] Jikes **Intermediate Code Representation**

File Format: Microsoft Powerpoint 97 - [View as HTML](#)
 The **Intermediate Representation** (IR) used by Jikes is a register based IR ... Finalizes the **exception** table. Converts **intermediate**-instruction offsets into ...
 www.cs.ualberta.ca/~amaral/courses/605-jit/jikesIR.ppt - [Similar pages](#)

All Packages from Compil3r.BytecodeAnalysis.* to Compil3r ...

BasicBlock: Represents a basic block in the quad **intermediate representation**. ... That is to say, a potential **exception** point does not end a basic block. ...
 www.docjar.com/docs/api/Compil3r/overview-summary.html - 9k - [Cached](#) - [Similar pages](#)

Known Problems and Workarounds

These object files contain an **intermediate representation** of the user code in a



USPTO

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☐ The ACM Digital Library ☒ The Guide

THE GUIDE TO COMPUTING LITERATURE
[Feedback](#) [Report a problem](#) [Satisfaction survey](#)
Terms used intermediate representation exceptionFound **65,128** of **869,947**

Sort results by

☒ [Save results to a Binder](#)
[Try an Advanced Search](#)

Display results

☒ [Search Tips](#)
[Try this search in The Digital Library](#)
☐ [Open results in a new window](#)

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐
1 [A single intermediate language that supports multiple implementations of exceptions](#)

Norman Ramsey, Simon Peyton Jones

 May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: [pdf\(900.75 KB\)](#)
 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We present mechanisms that enable our compiler-target language, C--, to express four of the best known techniques for implementing exceptions, all within a single, uniform framework. We define the mechanisms precisely, using a formal operational semantics. We also show that exceptions need not require special treatment in the optimizer; by introducing extra dataflow edges, we make standard optimization techniques work even on programs that use exceptions. Our approach clarifies the design s ...

2 [Efficient and precise modeling of exceptions for the analysis of Java programs](#)

Jong-Deok Choi, David Grove, Michael Hind, Vivek Sarkar

 September 1999 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 1999 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**, Volume 24 Issue 5
Full text available: [pdf\(1.16 MB\)](#)
 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

The Factored Control Flow Graph, *FCFG*, is a novel representation of a program's intraprocedural control flow, which is designed to efficiently support the analysis of programs written in languages, such as Java, that have frequently occurring operations whose execution may result in exceptional control flow. The FCFG is more compact than traditional CFG representations for exceptional control flow, yet there is no loss of precision in using the FCFG. In this paper, we introduce the FCFG r ...

3 [Engineering a customizable intermediate representation](#)

K. Palacz, J. Baker, C. Flack, C. Grothoff, H. Yamauchi, J. Vitek

 June 2003 **Proceedings of the 2003 workshop on Interpreters, virtual machines and emulators**
Full text available: [pdf\(322.87 KB\)](#)
 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

The Ovm framework is a set of tools and components for building language runtimes. We present the intermediate representation and software design patterns used throughout the framework. One of the main themes in this work has been to support experimentation with new linguistic constructs and implementation techniques. To this end, framework components were designed to be parametric with respect to the instruction set on which